CSE 259 - Logic in Computer Science
Fall 2024

Recitation-1

Waqar Hassan Khan

What is Prolog?

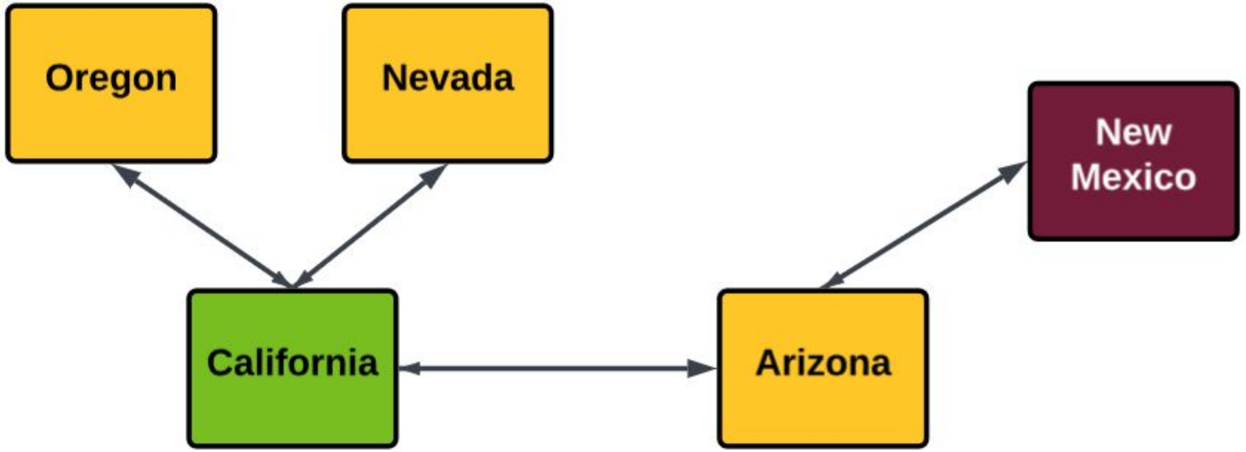
- Prolog is a **logic programming language**
- Prolog is intended primarily as a **declarative programming language**

Declarative Programming Language

Declarative programming is when you write your code in such a way that it describes what you want to do, and not how you want to do it. It is left up to the compiler to figure out the how.

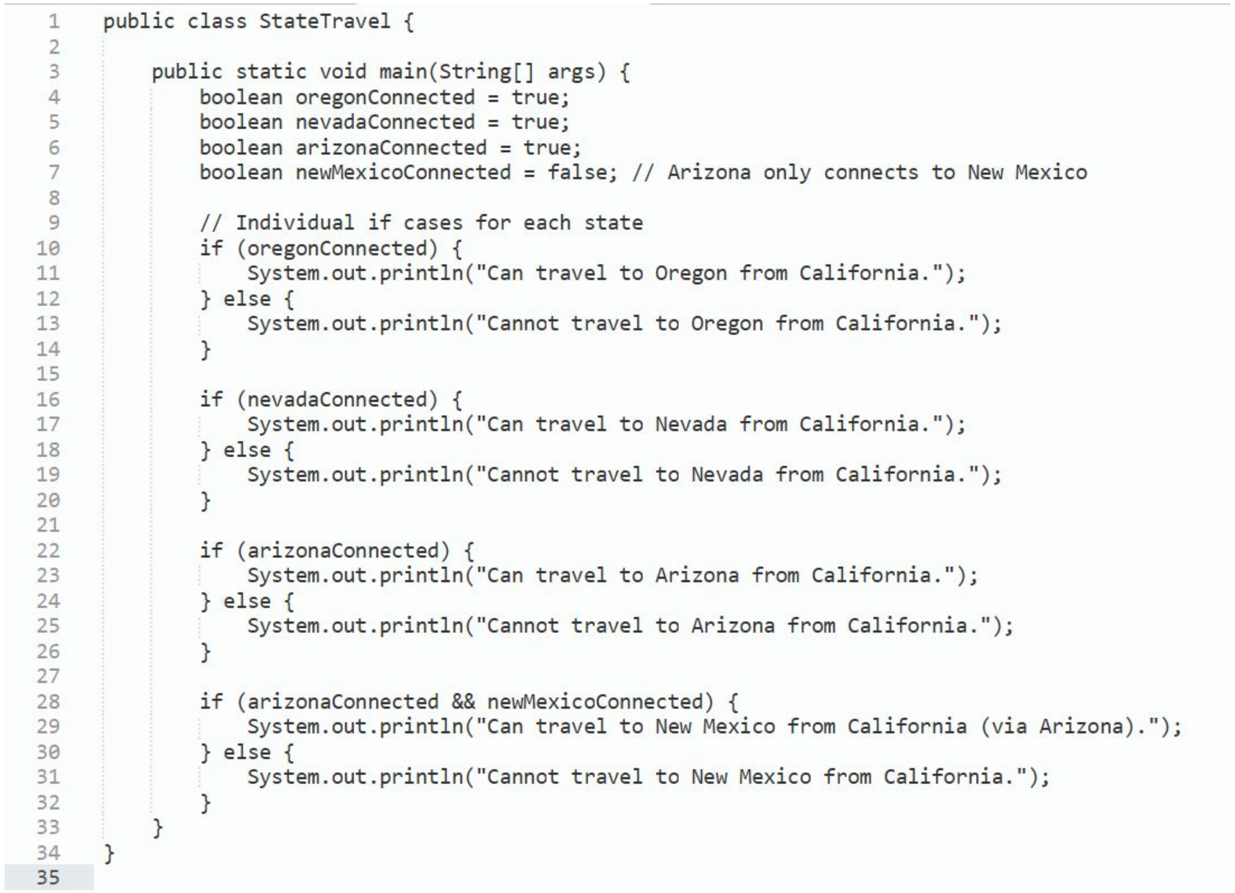
Example: SQL, Prolog (Obviously :D)

An example



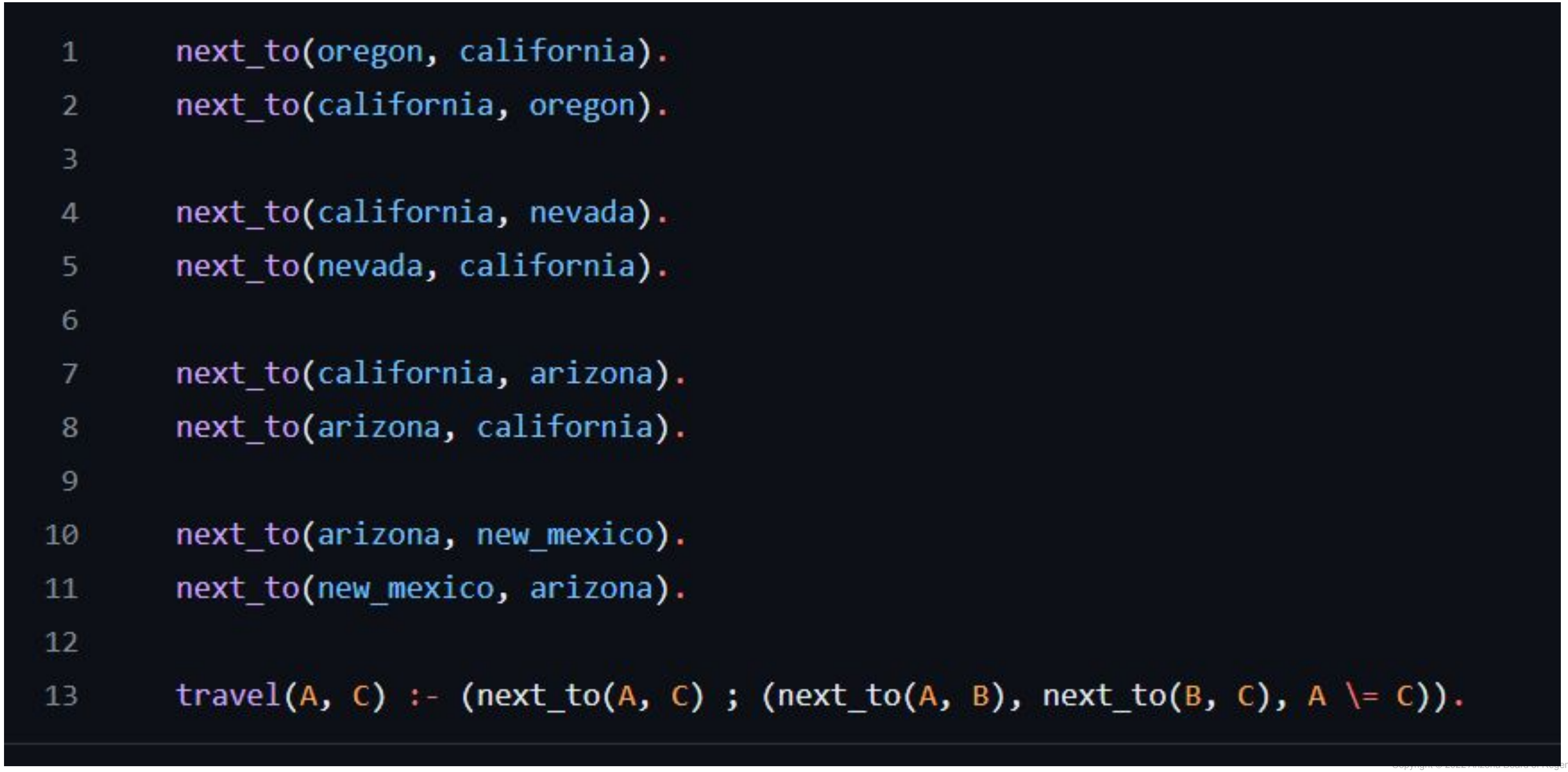
Suppose, we are in California. Which states can we visit from here? (We can travel if the states are adjacent)

Sample code for the example - Java



```java
public class StateTravel {

    public static void main(String[] args) {
        boolean oregonConnected = true;
        boolean nevadaConnected = true;
        boolean arizonaConnected = true;
        boolean newMexicoConnected = false; // Arizona only connects to New Mexico

        // Individual if cases for each state
        if (oregonConnected) {
            System.out.println("Can travel to Oregon from California.");
        } else {
            System.out.println("Cannot travel to Oregon from California.");
        }

        if (nevadaConnected) {
            System.out.println("Can travel to Nevada from California.");
        } else {
            System.out.println("Cannot travel to Nevada from California.");
        }

        if (arizonaConnected) {
            System.out.println("Can travel to Arizona from California.");
        } else {
            System.out.println("Cannot travel to Arizona from California.");
        }

        if (arizonaConnected && newMexicoConnected) {
            System.out.println("Can travel to New Mexico from California (via Arizona).");
        } else {
            System.out.println("Cannot travel to New Mexico from California.");
        }
    }
}
```

Sample code for the example - Prolog



```prolog
next_to(oregon, california).
next_to(california, oregon).

next_to(california, nevada).
next_to(nevada, california).

next_to(california, arizona).
next_to(arizona, california).

next_to(arizona, new_mexico).
next_to(new_mexico, arizona).

travel(A, C) :- (next_to(A, C) ; (next_to(A, B), next_to(B, C), A \= C)).
```

Basic Syntax

Term

- Basic building blocks of programs and data structures
- Similar to how variables, constants, and expressions are in other programming languages.

Basic Syntax

Different Terms

- **Variable:** starts with an uppercase letter or with an underscore. Example: **A, Ab, _a**
- **Constant :** atom or number. Atom starts with lowercase. Example: **john, apple, 23, 45, etc.**
- **Compound term :** formed by combining other terms using functors and parentheses. A functor is an atom that represents a function or relation symbol, and arguments are terms separated by commas and enclosed in parentheses. Example: **likes(john, mary). "likes"** is a **functor. "john"** and **"mary"** are atoms **(term!!)**

Basic Syntax

Predicate

- Fundamental concept used to define relations
- Represent statements or propositions that can be true or false
- Predicate name should be an **atom**
- There can be 0 or more arguments. Example: **green(apple)**, **capital_of(dhaka, bangladesh)**

Basic Syntax - Practice!

Which one of these are a variable?

1. X
2. y
3. _y
4. Fun

Basic Syntax - Practice!

Which one of these are a variable?

1. X
2. y
3. _y
4. Fun

Basic Syntax - Practice!

Which one of these are an atom?

1. X
2. y
3. _y
4. Fun

Basic Syntax - Practice!

Which one of these are an atom?

1. X
2. y
3. _y
4. Fun

Basic Syntax - Practice!

Which one of these are a predicate?

1. X
2. y
3. _y
4. Fun(car).
5. fun(Car)

Basic Syntax - Practice!

Which one of these are a predicate?

1. X
2. y
3. _y
4. Fun(car).
5. fun(Car)

Basic Syntax

Rule

- Contains **four** parts:

  Head, :-, Body, and a dot (.)

  fun(X) :- red(X), car(X).

  This example means - if X is a car and is red then it is fun.

- Symbols used:

  Implication **:-**

  Conjunction , **(and)**

  Disjunction ; **(or)**

Basic Syntax

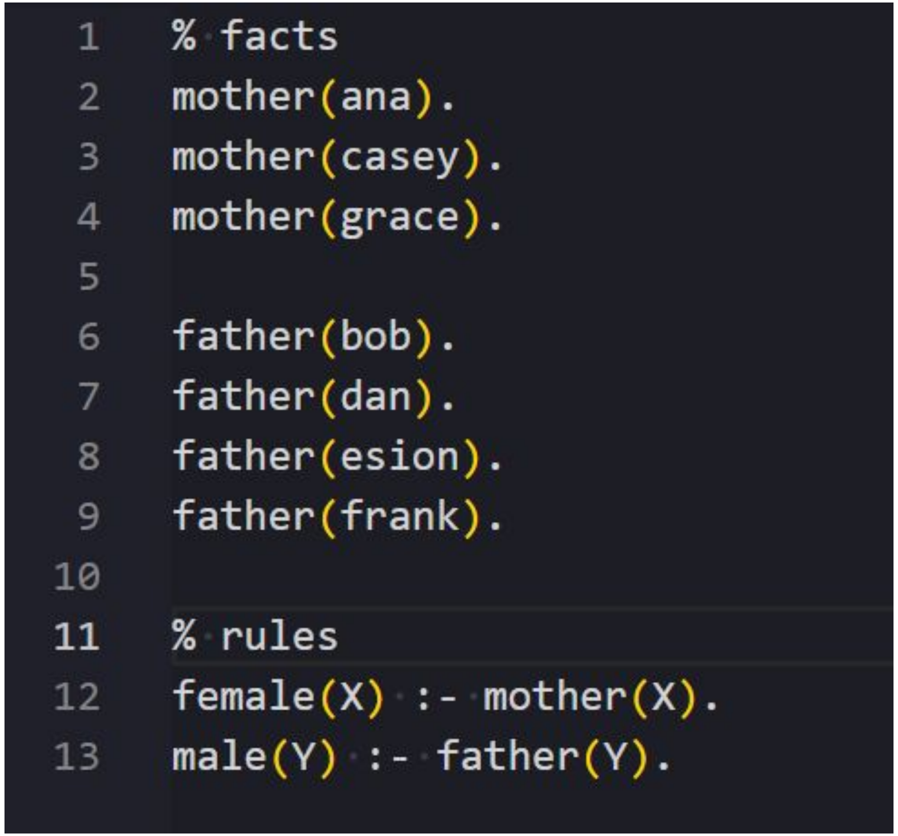
Facts

- Represents a relation between items
- Should **always begin with a lowercase letter** and **end with a full stop**. The facts themselves can consist of any letter or number

Lets solve a problem!

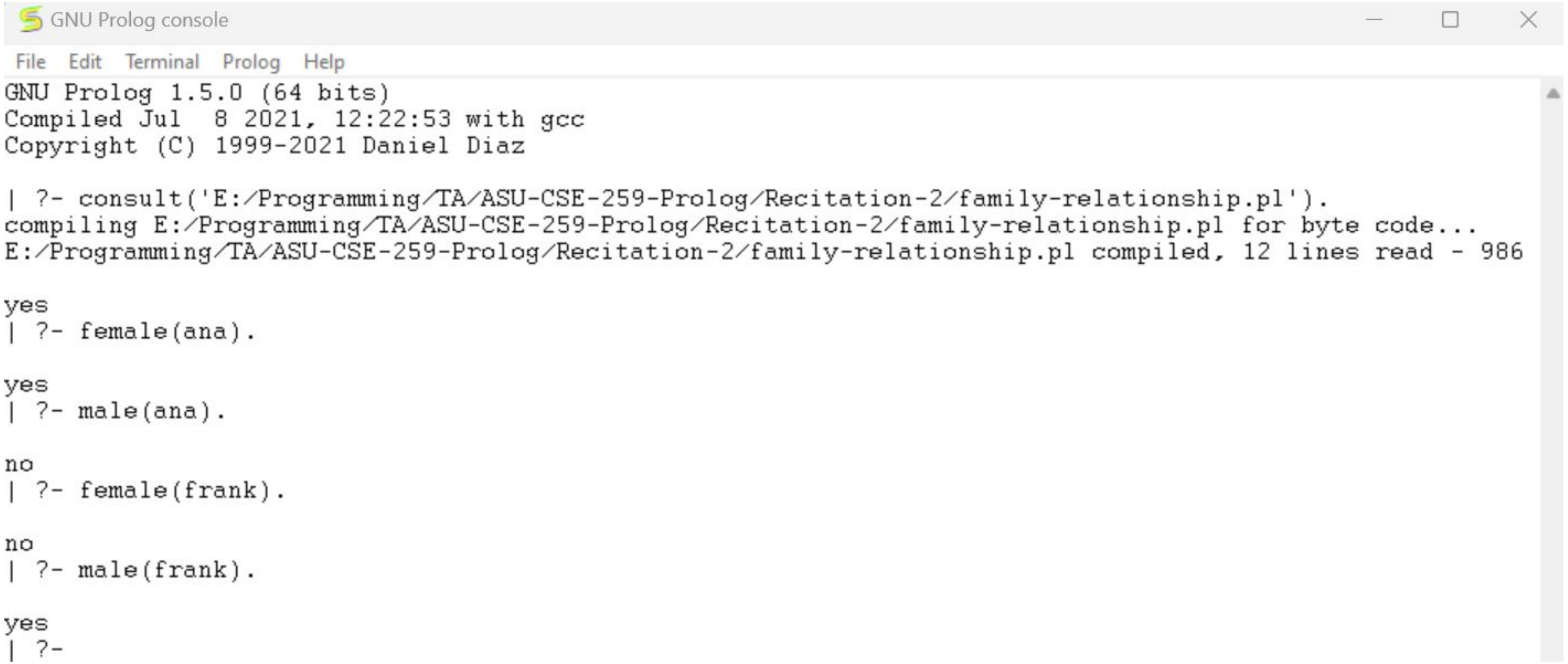
1. Define some facts
    - “ana”, “casey”, “grace” are mothers
    - “bob”, “dan”, “esion”, “frank” are fathers
2. Define two simple rules
    - If someone(X) is a mother then she is a female
    - If someone(Y) is a father then he is a male
3. Ask the following questions-
    - Is “ana” a female or male?
    - Is “frank” a female or male?

Lets solve a problem!



```
% facts
mother(ana).
mother(casey).
mother(grace).

father(bob).
father(dan).
father(esion).
father(frank).

% rules
female(X) :- mother(X).
male(Y) :- father(Y).
```

Lets solve a problem!



```
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- consult('E:/Programming/TA/ASU-CSE-259-Prolog/Recitation-2/family-relationship.pl').
compiling E:/Programming/TA/ASU-CSE-259-Prolog/Recitation-2/family-relationship.pl for byte code...
E:/Programming/TA/ASU-CSE-259-Prolog/Recitation-2/family-relationship.pl compiled, 12 lines read - 986

yes
| ?- female(ana).

yes
| ?- male(ana).

no
| ?- female(frank).

no
| ?- male(frank).

yes
| ?-
```

Lets solve a problem!

Running the code

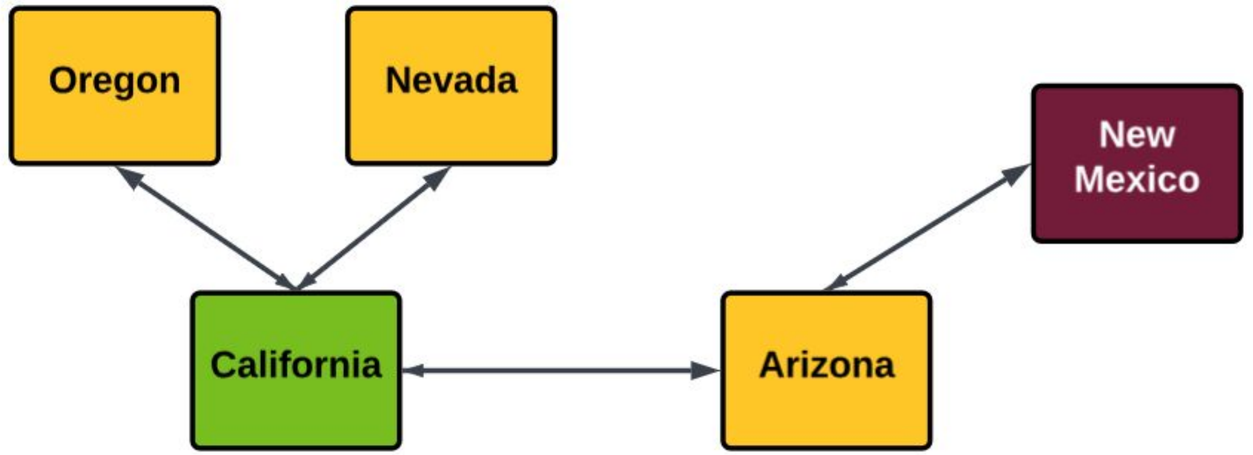
- Open Gnu Prolog -> File -> consult -> select the file

Understanding the result

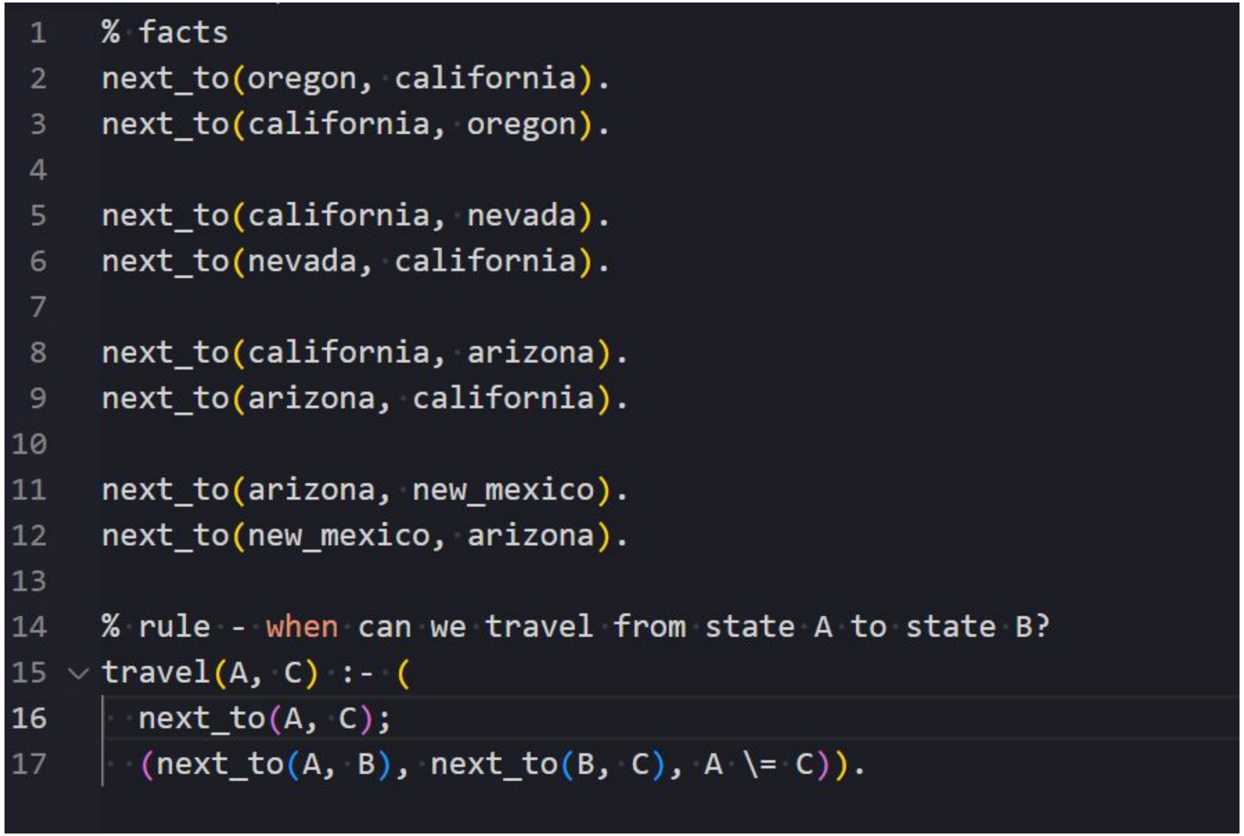
- Typically expect either a "yes" or a "no". Yes means there are one or more results. No means there's no solution

Lets solve another problem!

Where can we travel from California?

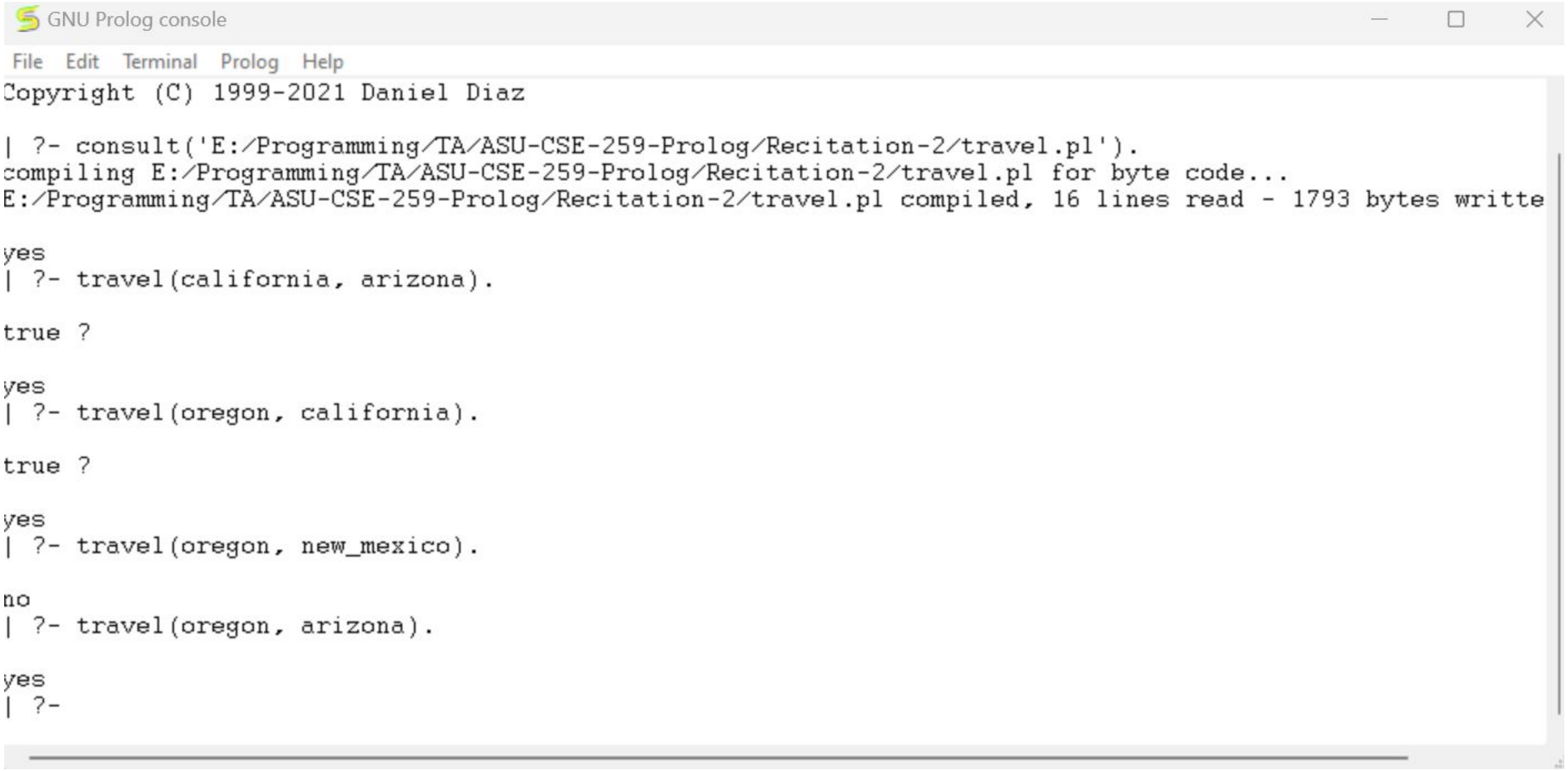


Lets solve another problem!



```
% facts
next_to(oregon, california).
next_to(california, oregon).

next_to(california, nevada).
next_to(nevada, california).

next_to(california, arizona).
next_to(arizona, california).

next_to(arizona, new_mexico).
next_to(new_mexico, arizona).

% rule - when can we travel from state A to state B?
travel(A, C) :- (
  next_to(A, C);
  (next_to(A, B), next_to(B, C), A \= C)).
```

Lets solve another problem!



```
Copyright (C) 1999-2021 Daniel Diaz

| ?- consult('E:/Programming/TA/ASU-CSE-259-Prolog/Recitation-2/travel.pl').
compiling E:/Programming/TA/ASU-CSE-259-Prolog/Recitation-2/travel.pl for byte code...
E:/Programming/TA/ASU-CSE-259-Prolog/Recitation-2/travel.pl compiled, 16 lines read - 1793 bytes writte

yes
| ?- travel(california, arizona).

true ?

yes
| ?- travel(oregon, california).

true ?

yes
| ?- travel(oregon, new_mexico).

no
| ?- travel(oregon, arizona).

yes
| ?-
```

Lets solve another problem!

Problem

- travel(oregon, new_mexico). did not work!